ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computer and System Sciences 70 (2005) 86–106

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Extracting the workflow critical path from the extended well-formed workflow schema☆

Jin Hyun Son[a,*], Jung Sun Kim[a], Myoung Ho Kim[b]

[a] *Department of Computer Science & Engineering, Hanyang University,1271 Sa-1 dong, Ansan, Kyunggi-Do 426-791, Republic of Korea*

[b] *Division of Computer Science, Korea Advanced Institute of Science and Technology, KAIST, 373-1 Kusung-dong, Yusung-gu, Taejon 305-701, Republic of Korea*

Received 7 June 2004

Available online 27 August 2004

Abstract

The critical path in a workflow schema is defined as the longest execution path from the start activity to the end activity. It can be utilized in many workflow issues such as workflow resource and time management. However, little work has been done on the critical path in a workflow because workflow control flows are much more complex than those represented with ordinary graphs and networks. In this paper, we first describe our workflow model with a set of workflow control constructs that provide sufficient power to express the models of most of today's business processes. Then, we propose a systematic method of identifying the critical path for a given workflow schema. Our proposed method is based on queuing theory because operational characteristics of the workflow schema can be modeled by a M/M/1 queuing network.

Keywords: Workflow; Critical path; Queuing network

☆ This work was supported in part by the Ministry of Information and Communications, Korea, under the Information Technology Research Center (ITRC) Support Program, and by the research fund of Hanyang University(HY-2003-T).

* Corresponding author.

E-mail addresses: jhson@cse.hanyang.ac.kr (J.H. Son), jskim@cse.hanyang.ac.kr (Jung Sun Kim), mhkim@dbserver.kaist.ac.kr (Myoung Ho Kim).

0022-0000/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2004.07.001

1. Introduction

A workflow is defined as "an automated procedure where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal" [11]. It is composed of activities that are interconnected by workflow control flows. A workflow schema, commonly abstracted with a network of activities, is a description of a workflow. A workflow management system (WFMS) completely defines, manages and processes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic [11,14]. An execution of a workflow is called a workflow instance, simply instance. Multiple instances are generally created for the same workflow. Nowadays, the concept of a workflow attracts a great interest in the respect of automating and computerizing business processes in whole or part, which gives us many benefits such as structural efficiency of a business process, performance improvement, flexibility, better process control, improved customer service, etc. Fig. 1 is an example of the new service provisioning workflow in a telecommunication company.

A workflow schema can contain several alternative execution paths from the start activity to the end activity. In particular, the longest execution path is called the critical path. Hence, the execution time of the entire workflow process is dominated by the critical path. Finding the critical path gives us important information of a workflow schema. More than anything else, the critical path may contain many workflow bottleneck points. Therefore, high-performance workflow systems, one of the most outstanding workflow issues, can be achieved by efficiently managing the critical path. Time-constrained workflows can also be its noticeable application area. Since many business processes that are abstracted to workflow schemas have time constraints such as deadlines, time management functionality should be provided to control the lifecycle of workflow instances. If a workflow instance violates the activity deadline during workflow execution, the workflow management system may escalate the instance. The effects of an escalation depend on the semantics of the activity that missed its deadline. Usually, escalations contain the execution of additional activities, the compensation of finished activities, or human intervention. Because they increase the operational costs of business processes, it is desirable to reduce the number of escalated workflow

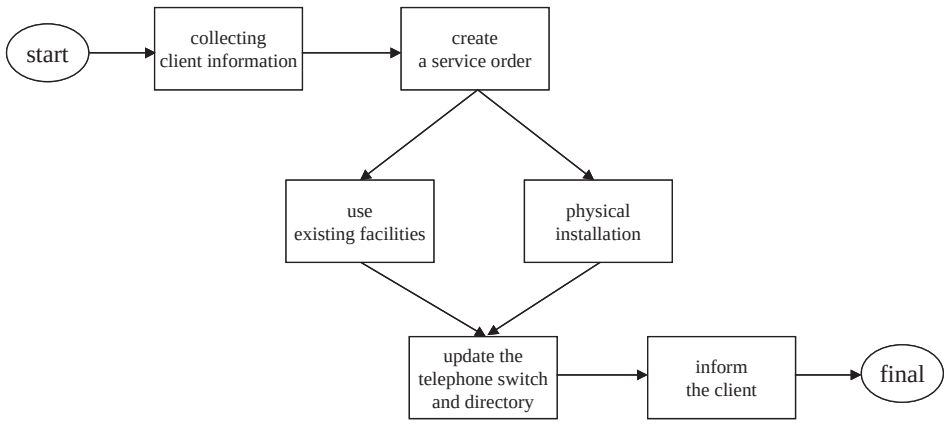


Fig. 1. A workflow schema for new service provisioning.

executions as well as their operational costs associated with escalations. If we properly use the concept of the critical path in this area, effective time management methods can be developed.

The critical path in a workflow can be dynamically or statically determined according to its decision points. During workflow run-time, the critical path determined dynamically may change every decision time, based on the current state of a workflow system. However, the dynamic decision of the critical path may impose considerable amount of workloads on the workflow system. On the other hand, the static critical path that is considered in this paper is determined during workflow build-time. Because we can predict the future workflow execution environment by using the statistics collected during the past executions, the static critical path can be utilized in many workflow research areas, including the resource allocation and the resource management policy. In addition, the process of identifying the static critical path does not give any burden to workflow systems.

At the time of this writing, the workflow reference model defined in Workflow Management Coalition (WfMC) specifies six basic workflow control constructs: sequencing, conditional branching (i.e., OR-Split), asynchronous join (i.e., OR-Join), split parallelism (i.e., AND-Split), join synchronization (i.e., AND-Join), and iteration [11]. These constructs can make sequential, alternative, parallel, and iterative workflow control structures that have been used to define workflow schemas in most previous workflow research [4,12,16,19,21,22]. Here, a sequential and an iterative workflow control structure are formed by a sequencing and an iteration construct, respectively. An alternative workflow control structure is made by combining OR-Split and OR-Join. And, a parallel workflow control structure is built by combining AND-Split and AND-Join. However, the workflow control flows represented by these basic workflow control constructs are not enough to cover today's complex business logic. For example, let us consider the review process of a paper [1]: "A paper needs to be sent to three external reviewers. Upon receiving two reviews, the paper can be processed and the third review can be ignored." This review process cannot be supported by the basic workflow control constructs mentioned above. To provide sufficient power in expressing the models of complex business processes, we introduce a few useful workflow control constructs. Note that the aforementioned review process of a paper can be supported by the combination of AND-Split and Selective-AND-Join control constructs that will be specified in Section 2.

We propose an efficient method to find out the critical path in a workflow schema that is built by the extended workflow control constructs defined in this paper. Up to now, there is no general agreement to a complete set of workflow control constructs. The growing complexities of business processes may require new advanced workflow control constructs. Because our proposed method is extensible, we can easily encompass newly agreed workflow control constructs to support future business logic.

Program Evaluation and Review Technique/Critical Path Method (PERT/CPM) is a well-known network analysis technique that has been applied to many applications such as research and development, construction programs, programming computers, maintenance planning, and so on. PERT/CPM has several outstanding advantages, especially clarity of presentation, cost saving, critical path analysis, and time control [2,8,23]. However, its usage requires some constraints: "All activities must be completed in order for the project to be completed. All activities leading to an event must be completed before the event can be realized. And, there is no looping allowed, that is, no event may be repeated [2,20,26]." The precedence diagram method (PDM) extends PERT/CPM by using lead-lag factors to overcome one type of precedence requirement (i.e., finish-to-start) of PERT/CPM [13,24]. In other words, while PERT/CPM allows an activity to be performed only after its immediate predecessors are finished, PDM introduces additional precedence relationships such as finish-to-finish, start-to-start, and start-to-finish. The lead-lag relationships in PDM may also be expressed as percentages, such as task A cannot start unless at least 25%

of task B is completed. And, Graphical Evaluation and Review Technique (GERT) was also developed to overcome many of the limitations of PERT/CPM, while still retaining the characteristic of ease in system modeling [2,20,24]. GERT permits the looping of an activity and the bypassing of an activity with two types of branching, i.e., deterministic and probabilistic.

In a workflow area, we cannot, however, use the previous above-mentioned methods in deciding the critical path of a workflow due to the following two main reasons: One is that they are not extensible enough to support all defined or newly agreed workflow control constructs. The other is that they assume the existence of a single execution instance for a project process at a time. As multiple instances can be concurrently executed for a workflow schema, some instances should wait at the queue of an activity while other instances that arrive ahead are served. Hence, the average execution time of a workflow instance in an activity is computed by the average service time of the activity plus the average waiting time at the queue of the activity. In many cases, the average execution time is dominated by the average waiting time rather than the average service time. In consequence, the appropriate queuing analysis is required to manage the waiting time at the queue, like our proposed method.

There is some research for workflow resource management. Son and Kim [21] discussed a method to improve the performance of time-constrained workflow processing by increasing the processing capacities for certain activities. An efficient task allocation method in a distributed workflow environment was proposed in [15]. The primary objective of this method is to reduce the processing costs of a workflow using the spatial locality principle.

The general workflow time management issues such as computing activity deadlines, checking time constraints, and monitoring escalations are mentioned in [5,6,18]. Especially, Panagos and Rabinovich [17] introduced the concept of a predictive workflow with the idea that it is preferable to escalate workflow instances as early as possible if the escalation is unavoidable. Eder et al. [4] presented a CPM-based framework for computing activity deadlines so that the overall process deadline is met and all external time constraints are satisfied. Panagos and Rabinovich [16] studied how to efficiently manage dynamic activity deadlines, which is based on the deadline assignment methods considered in a distributed real-time software environment [9]. Son et al. [22] insisted on the importance of static deadline allocation in a workflow in comparison with dynamic deadline allocation of Panagos abd Rabinovich [16]. And, Kafeza and Karlapalem [7] proposed a framework (T-WfMC) for incorporating time-management information in workflow execution and specification.

The remainder of the paper is organized as follows: Section 2 describes the workflow model considered in the paper. Section 3 proposes our method that systematically determines the critical path in a workflow schema, and its overall example is provided in Section 4. After we discuss some considerations in Section 5, we conclude the paper with its contribution and further work in Section 6.

2. Workflow model

2.1. Workflow control constructs

A workflow schema is represented with a set of nodes and directed edges. It begins from the start node and ends with the final node. Each node denotes a workflow activity, and each directed edge denotes a transition between two nodes, i.e., a branch of execution. A workflow control construct defines a routing from a set of nodes S to a set of nodes D in a workflow schema. S and D are called a source and a

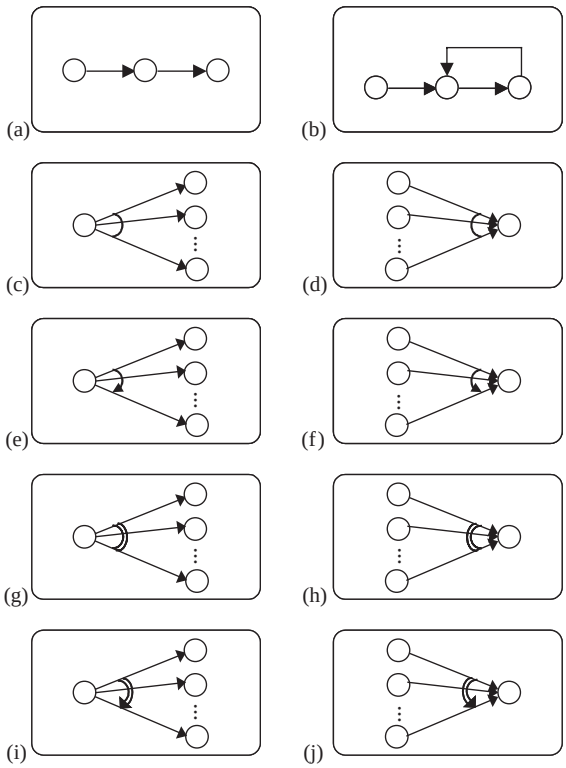


Fig. 2. Workflow control constructs. (a) *SEQUENCE*, (b) *LOOP*, (c) *AND-split*, (d) *AND-join*, (e) *Selective-AND-split*, (f) *Selective-AND-join*, (g) *OR-split*, (h) *OR-join*, (i) *Preference-OR-split* and (j) *Preference-OR-join*.

destination of the routing, respectively. A routing is denoted by a set of directed edges with appropriate conditions, if needed. We can classify workflow control constructs into four categories, i.e., sequence, loop, split and join. Let $|S|$ and $|D|$ denote the number of nodes in a set S and D, respectively. Then, both a sequence and a loop control construct take $|S| = |D| = 1$. A split control construct takes $|S| = 1$ and $|D| > 1$. And, a join control construct takes $|S| > 1$ and $|D| = 1$. Fig. 2 shows graphical expressions of the workflow control constructs considered in this paper.

In a sequence control construct, activities are executed in order under a single thread of execution, which means that the succeeding activity cannot start until the preceding activity is completed. A loop control construct makes one or more activities executed iteratively until a condition is met.

A split control construct is defined as a flow controller that makes a single execution thread proceed to one or more concurrent threads according to the controller's control context. There are four split control constructs: AND-split, Selective-AND-split, OR-split, and Preference-OR-split. By an AND-split, a single thread of execution splits into multiple threads that are executed in parallel. Here, all the activities in the destination D of an AND-split are executed concurrently. In a Selective-AND-split (often called a N-out-of-M split), one or more activities in the destination are selected for concurrent execution. When all the activities in the destination are selected, it degenerates into the AND-split. On the other hand, in an OR-split, only a single execution thread of control can proceed among multiple alternative workflow branches.

In a Preference-OR-split, each branch has a preference. When there are several branches satisfying their transition conditions, a branch with the highest preference is selected to get an execution thread.

A join control construct also plays a role of a flow controller, where one or more execution threads converge into a single activity. The class of the join control construct is composed of an AND-join, a Selective-AND-join, an OR-join, and a Preference-OR-join. An AND-join makes two or more parallel execution threads synchronized into a single common thread with a condition that the number of parallel execution threads must be equal to that of workflow branches joined. A Selective-AND-join, being different from an AND-join, selects one or more incoming branches and ignores others. The parallel threads going through the selected branches are synchronized at the join point before the next activity begins. By an OR-join, two or more alternative branches re-converge to a single common activity. As no parallel activity execution has occurred at the join point, no synchronization is required. In a Preference-OR-join where each incoming branch to the destination has a preference, multiple threads of execution can arrive at the destination. However, only one thread of execution can be accepted, depending on the preference on each branch. Thus, it is similar to the OR-join in that both control constructs accept only one execution thread. And, it has no synchronization among the incoming threads of execution, which is different from the Selective-AND-join. A Preference-OR-join may typically select one with the highest preference.

Note that any split control construct must be combined with its matching join control construct to be a complete control flow. Hence, we define the concept of the closure.

Definition 2.1. For a split control construct α, there is a set of join control constructs C_α that can be combined with α, where the combination can produce a meaningful flow of control. Then, this set C_α is called the *closure* of α.

Let C_α denote the closure of α.

$$C_{AND-split} = \{AND - join, Selective - AND - join, Preference - OR - join\}$$
$$C_{Selective-AND-split} = \{Selective - AND - join, Preference - OR - join\}$$
$$C_{OR-split} = \{OR - join, Preference - OR - join\}$$
$$C_{Preference-OR-split} = \{OR - join, Preference - OR - join\}.$$

As an example, the pair (AND-split, Preference-OR-join) as in Fig. 3 can be used when we prefer a first booked airline while making reservations to three different airlines at the same time. The other two reservations will be canceled.

Definition 2.2. Any subnetwork in a work flow schema that satisfies one of the following conditions is called a *non-sequential control block* (simply *control block*).

- A subnetwork that starts from a split control construct and ends with its matching join control construct.

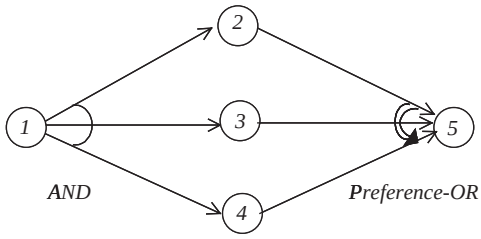


Fig. 3. AND-split & Preference-OR-join.

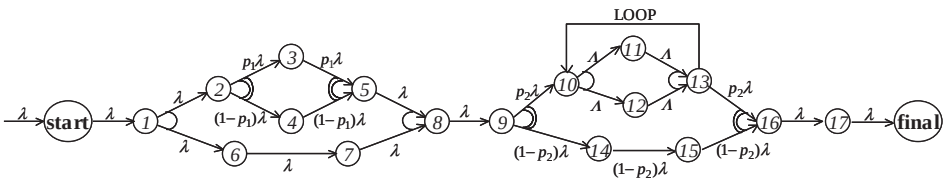


Fig. 4. Workflow queuing network.

- A subnetwork that consists of all the nodes and edges between the source and the destination of the LOOP.

Note that a sequence control construct is not a control block from Definition 2.2. Let a control block *V* be part of another control block *U*. Then, we call "*U* completely contains *V*" if all the nodes and edges of *V* are within *U*. Otherwise, "*U* partially contains *V*". Any single control block that neither completely nor partially contains any other control block is called an *atomic control block*. When a control block *U* completely contains a control block *V* and does not partially contain any other control block, *U* is called a *nested control block*. In this case, *U* is an outer control block (shortly outer block) of *V*, and *V* is an inner control block (shortly inner block) of *U*. Let there be no control block between *U* and *V*. Then, *U* is an immediate outer block of *V*, and *V* is an immediate inner block of *U*. Note that for any control block *U*, there can be multiple immediate inner blocks. However, there can be only one immediate outer block. A control block that is not contained in any other control block is called a *top-level control block*. An atomic control block or a nested control block can be a top-level control block. A node that is not contained in any control block is called a *simple node*. As a result, a workflow schema is a sequence of top-level control blocks and simple nodes as in Fig. 4. The workflow schema of Fig. 4 is composed of two top-level control blocks and one simple node (i.e., activity 17).

If a workflow schema is designed without any discipline for workflow control constructs, it may have some errors that are difficult to be found and corrected [16,21]. For example, if the destination of the loop's feedback is activity 5 instead of activity 10 in Fig. 4, the workflow schema does not have correct control flows due to the context of an AND-join at activity 8. There are structural and semantic workflow errors. The structural correctness of our workflow is based on two main properties that are originated from a structured programming language such as Pascal, C, or Java: "One is that workflow control structures built by sequence, iteration, or the matching (split construct, join construct) pairs are all single-entry and single-exit. The other is that complex control flows can be made by nesting them." Hence, our workflow

can be said to have the same structural correctness as a structured programming language. On the other hand, the semantic workflow errors such as reachability and deadlock-free routing can be probed using the workflow control information of pre/post-activity conditions and transition conditions between activities. Because most of such information can be determined at workflow run-time, we cannot detect and eliminate all semantic errors from syntactic workflow definitions. Because a well-formed workflow defined in the following is based on the concepts of the closure and control block, it is, therefore, free from workflow structural errors.

Definition 2.3. An atomic control block and a nested control block are called *well-formed control blocks*. A workflow schema built by well-formed control blocks and simple nodes is called a *well-formed workflow schema*.

From now on, all workflow schemas mentioned in this paper denote well-formed workflow schemas if there is no ambiguity.

2.2. Workflow queuing network

Numerous natural physical and organic processes exhibit behavior that is probably meaningfully modeled by Poisson processes. An important application of the Poisson distribution arises in connection with the occurrence of events of a particular type over time. The exponential distribution is frequently used as a model for the distribution of times between the occurrence of successive events such as customers arriving at a service facility. Because of them, the Poisson process and the exponential distribution have been used to analyze many areas of computer engineering [10]. Hence, we consider in this paper that the service requests for a workflow arrive by a Poisson process and each activity agent has exponential service time. As a result, we can model a well-formed workflow schema as a M/M/1 queuing network such as telephone networks or computer communication networks, with the following assumptions. This allows us to analyze some important workflow properties.

Assumption 2.1. Queue discipline in an activity is first come-first served (FCFS), i.e., workflow instances are served in the same order in which they arrive.

Assumption 2.2. The queue size of an activity is sufficiently large to accommodate a large number of workflow instances. Namely, workflow instances can wait the service of an activity for a long time.

In a M/M/1 workflow queuing network, each activity is an independent M/M/1 queuing system [25]. We can, therefore, specify the arrival and departure rate in each activity as in Fig. 4, based on the facts mentioned below. We basically know the initial request rate to the start node, the service rate in each activity, and the branch selection-probabilities in each workflow control construct.

In a Sequence control construct, if the arrival process of an activity is a Poisson process, its departure process is also the same Poisson process [10]. The decomposition and superposition of independent Poisson processes in the OR-split/join and Preference-OR-split/join are known to be also Poisson processes from time reversibility [25]. Activity 2, 5, 9, and 16 in Fig. 4 are the examples. Even though the actual internal flow of a LOOP is not a Poisson process due to its feedback, it is already known that the LOOP

behaves as if its internal activities are independent M/M/1 systems [3]. The departure processes of an AND-split and a Selective-AND-split (e.g., activity 1 in Fig. 4) are clearly Poisson processes if their arrival processes are Poisson. However, the arrival processes by an AND-join and a Selective-AND-join (e.g., activity 8 in Fig. 4) are not actually Poisson processes because parallel execution threads must be synchronized. Since a point of synchronized time by these control constructs is usually determined by the path with the longest average execution time among n workflow branches along which as many threads are executed in parallel, the arrival process can be approximated by the Poisson process of the requests going through the longest execution path. After all, a workflow schema can be modeled by a M/M/1 queuing network in which each activity is an independent M/M/1 system with Poisson arrival and departure processes.

3. Critical path

3.1. WCP algorithm

The critical path is defined as the longest execution path from the start node to the final node in a workflow schema. We propose a method called the Workflow critical path (WCP) that determines the critical path in a well-formed workflow schema modeled by a M/M/1 queuing network. Since many workflow instances for the same workflow schema can be concurrently executed, some workflow instances should wait at the queue of an activity while other workflow instances that arrive ahead are served. This means that *the average execution time* of a workflow instance in an activity is *the average service time* of the activity plus *the average waiting time* at the queue of the activity.

It is clear that since a workflow schema is a sequence of simple nodes and/or top-level control blocks, all the simple nodes and the longest execution path in each top-level control block are parts of the critical path. Therefore, WCP is essentially a method to find out the longest execution path (called the critical sub-path) in each top-level control block and then combine the critical sub-paths to form the critical path.

To determine the critical sub-path in a top-level control block, the longest execution path is selected from the innermost control block to the outermost control block, which is for step 2 of WCP algorithm in Fig. 5. Note that all innermost control blocks are atomic control blocks. If the innermost control block is a LOOP, the LOOP is transformed into its corresponding sequence control construct, which is for step 4 in the algorithm. If the innermost control block consists of a split and a join control construct, a path with the longest average execution time in the control block is selected and then transformed into a single activity, which comes under step 5 (The necessity for step 5.2, i.e., “Transform the path into a single activity” will be explained in Section 3.7). At the exit of the WHILE loop in the algorithm, a new atomic control block corresponding to each top-level control block is generated. And then, the longest execution path is determined in the newly generated atomic control block as in step 6. This path is the critical sub-path of the top-level control block and becomes part of the critical path. Finally, the critical path is determined by combining all the critical sub-paths with simple nodes. Fig. 6 depicts all the steps to find out the critical sub-path.

In the next sections (from Sections 3.2 to 3.7), we present how to solve WCP steps mentioned above, namely, steps 4–6 in Fig. 5. From now on, a control block consisting of a α-split and a β-join control construct is denoted by a (α,β) split/join control block, shortly (α,β) control block.

```
1. All the simple nodes belong to the critical path.
2. For (each top-level control block)
  {
    3. WHILE (the top-level control block is a nested control block)
      /* At the exit of this WHILE loop, the nested control block becomes
       an atomic control block */
      {
        /* When there are multiple innermost control blocks, the order of
         processing among innermost control blocks does not matter. */
        4. IF (its innermost control block is a LOOP atomic control block)
          4.1 Transform the LOOP into a sequence control construct.
        5. ELSE IF (its innermost control block is an atomic control block
          consisting of a split and a join control construct)
            5.1 Select a path with the longest average execution time
              in the control block.
            5.2 Transform the path into a single activity.
      }
    /* Here, the top-level control block is an atomic control block */
    6. Determine the longest execution path in the control block
      which is the critical sub-path of the top-level control block.
  }
7. Combine all the critical sub-paths with simple nodes.
  The resulting path becomes the critical path of a workflow schema.
```

Fig. 5. WCP algorithm.

3.2. A LOOP atomic control block

Since we mentioned in Section 2.2 that each activity in a LOOP atomic control block can be considered as an independent M/M/1, the LOOP can be transformed into its corresponding sequence control construct as in Fig. 7, which is for step 4.1 in the WCP algorithm. Because of the feedback in a LOOP, the arrival rate Λ of the LOOP is stated as

$$\Lambda = \lambda + (1 - p)\Lambda,$$

$$\Lambda = \frac{\lambda}{p},$$

where $1 - p$ is the probability that the feedback occurs in node n.

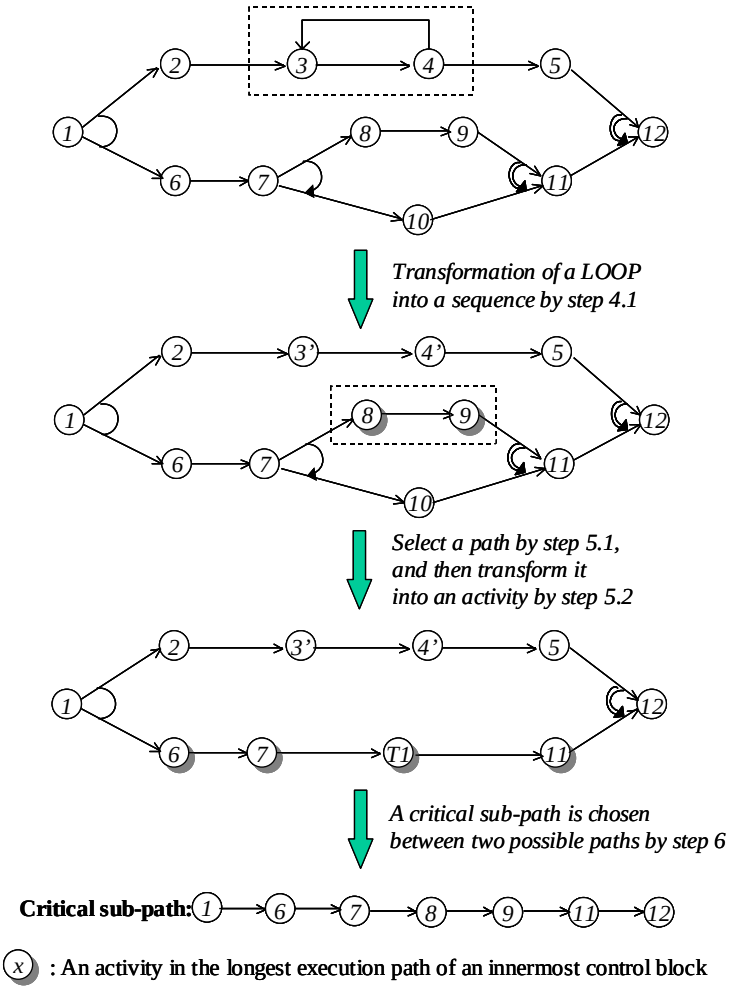


Fig. 6. Finding the critical sub-path.

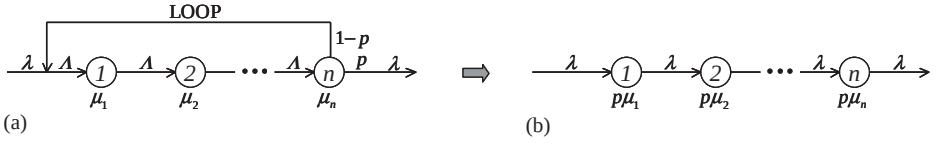


Fig. 7. Transformation of a LOOP atomic control block.

The average number of service requests waiting in front of activity i in Fig. 7(a) is $\frac{\rho_i}{1-\rho i}$ where ρ_i is $\frac{\Lambda}{\mu_i} = \frac{\lambda}{p\mu_i}$, $i = 1, \ldots, n$. Hence, the average execution time of the LOOP from the Little's formula is

$$\left(\frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2} + \cdots + \frac{\rho_n}{1-\rho_n}\right)\frac{1}{\lambda}$$

$$\frac{1}{p\mu_1 - \lambda} + \frac{1}{p\mu_2 - \lambda} + \cdots + \frac{1}{p\mu_n - \lambda}$$

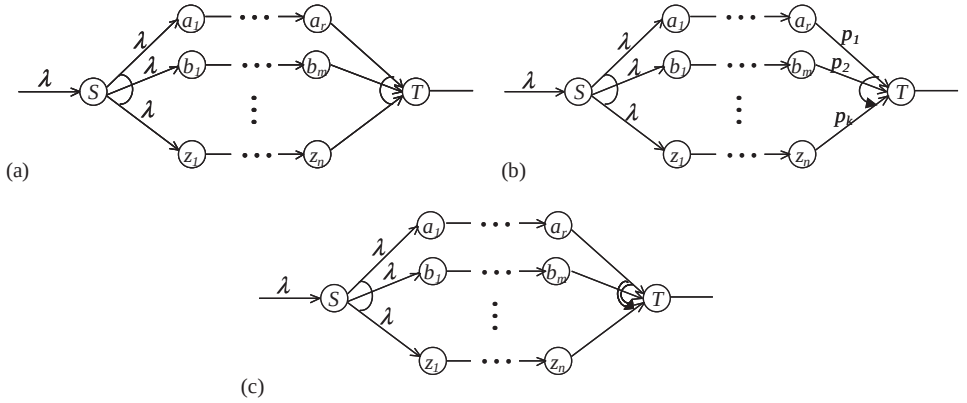


Fig. 8. The closure of the AND-split: (a) AND-split & AND-join, (b) AND-split & Selective-AND-join and (c) AND-split & Preference-OR-join: λ: arrivalrate, p_t: probability to be selected.

that is equal to the average execution time of the sequence composed of n activities with the arrival rate λ and service rate $p\mu_i$ of activity i as in Fig. 7(b). Here, the average execution time in a M/M/1 activity is computed by $\frac{1}{\mu-\lambda}$ where μ is the service rate and λ is the arrival rate. A LOOP atomic control block can, therefore, be transformed to a sequence control construct.

The following Sections 3.3 to 3.6 is for step 5.1 in the WCP algorithm, i.e., managing a control block consisting of (a split control construct, a join control construct).

3.3. The closure of the AND-split control construct

The closure of the AND-split is {*AND–join*, *Selective–AND–join*, *Preference–OR–join*}. Each node in the destination of an AND-split, i.e., $a_1, b_1, \ldots, z_1$ in Fig. 8 has the same arrival rate λ as that of the split activity S.

In a (AND-split, AND-join) atomic control block as in Fig. 8(a), the longest execution path is a path whose total average execution time is $MAX(\sum_i \frac{1}{\mu_i-\lambda})$, where λ is the arrival rate and μ_i is the service rate of activity i in the path. This is caused by the fact that the average execution time W of an activity modeled by the M/M/1 queuing system is $W = \frac{1}{\mu-\lambda}$ where μ is the service rate and λ is the arrival rate in the activity.

Consider a (AND-split, Selective-AND-join) atomic control block as in Fig. 8(b), where p_j is the selection probability of incoming branch j to the destination of the Selective-AND-join. Let $path_j$ denote the path that goes through this branch j. Here, we can note that all the nodes in $path_j$ behave as if they have arrival rate $p_j\lambda$ by the Selective-AND-join. Hence, the longest execution path in this control block is a path whose total average execution time is $MAX(\sum_i \frac{1}{\mu_i-p_j\lambda})$, where $p_j\lambda$ is the arrival rate of all the nodes in $path_j$ and μ_i is the service rate of activity i in the path.

Consider a (AND-split, Preference-OR-join) atomic control block as in Fig. 8(c). If we assume that the preference of a Preference-OR-join is to select an incoming branch through which the thread of execution arrives first, the longest execution path in this control block is a path whose total average execution time is $MIN(\sum_i \frac{1}{\mu_i-\lambda})$, where λ is the arrival rate of all the nodes in $path_j$ and μ_i is the service rate of

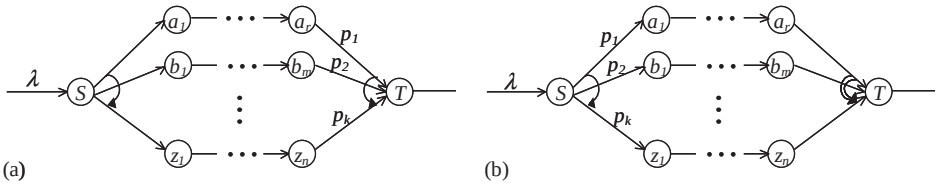


Fig. 9. The closure of the Selective-AND-split: (a) Selective-AND-split & Selective-AND-join and (b) Selective-AND-split & Preference-OR-join.

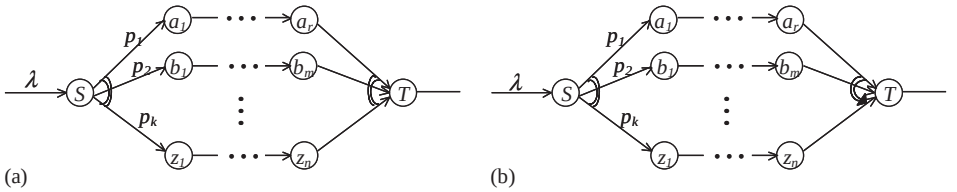


Fig. 10. The closure of the OR-split: (a) OR-split & OR-join and (b) OR-split & Preference-OR-join.

activity i in the path. The expression may be changed according to how an incoming branch is selected in a Preference-OR-join.

3.4. The closure of the selective-AND-split control construct

The closure of the Selective-AND-split is {*Selective–AND–join*, *Preference–OR–join*}.

In a (Selective-AND-split, Selective-AND-join) atomic control block as in Fig. 9(a), because path selection is determined by the Selective-AND-join, the longest execution path in the control block can be determined in a way similar to the (AND-split, Selective-AND-join) control block.

In a (Selective-AND-split, Preference-OR-join) atomic control block as in Fig. 9(b), we have outgoing branch selection probability p_i by the Selective-AND-split. Let us also assume that a Preference-OR-join selects an incoming branch through which the thread of execution arrives first. Then, the longest execution path in the control block is a path whose total average execution time is $MIN(\sum_i \frac{1}{\mu_i - p_j\lambda})$, where $p_j\lambda$ is the arrival rate of outgoing branch j from the source of the Selective-AND-split and μ_i is the service rate of activity i in the path.

3.5. The closure of the OR-split control construct

The closure of the OR-split is {*OR–join*, *Preference–OR–join*}. By the control context of the OR-split, each outgoing branch has exclusive selection probability p_i with $\sum\ p_i = 1$. Because the OR-split selects only one outgoing branch, both a (OR-split, OR-join) control block and a (OR-split, Preference-OR-join) control block as in Fig. 10 have the same way by which they find out the longest execution path. Namely, the longest execution path is a path whose total average execution time is $MAX(\sum_i \frac{1}{\mu_i - p_j\lambda})$, where $p_j\lambda$ is the arrival rate of outgoing branch j and μ_i is the service rate of activity i in the path.

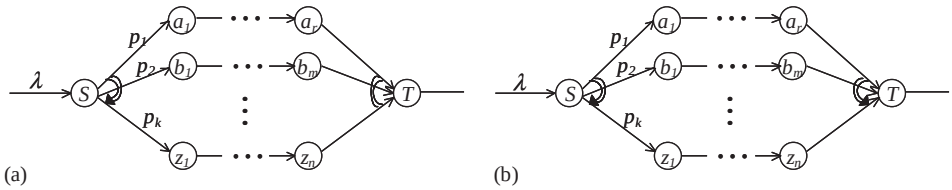


Fig. 11. The closure of the preference-OR-split: (a) Preference-OR-split & OR-join and (b) Preference-OR-split & Preference-OR-join.

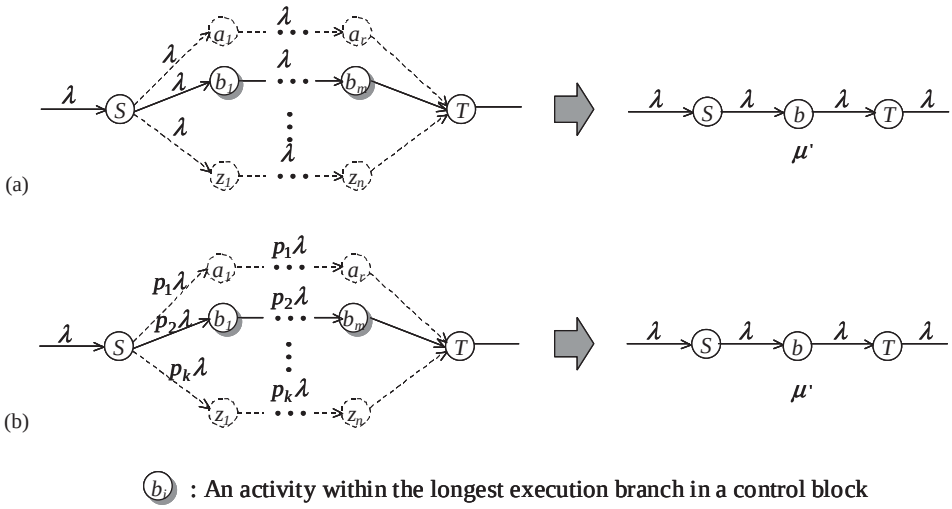


Fig. 12. Transformation of a sequence.

3.6. The closure of the preference-OR-split control construct

The closure of the Preference-OR-split is $\{OR - join, Preference - OR - join\}$ as depicted in Fig. 11. The Preference-OR-split has similar control context with the OR-split in that they all select only one outgoing branch with probability p_i. The longest execution path, therefore, can be computed in the same way as the closure of the OR-split.

3.7. A sequence control construct

After finding out the longest execution path in a control block (step 5.1 in the WCP algorithm), we transform the sequence path into a single activity *b* as in Fig. 12, which is for step 5.2 of the algorithm. Especially, because the arrival rate of the selected $path_j$ in Fig. 12(b) is different from that of activity *S*, the newly generated activity *b* must have the same arrival rate as activity *S*. In consequence, we in this transformation determine the service rate μ' of activity *b* satisfying that the average execution time of activity *b* is equal to that of the sequence, while keeping the arrival rate of activity *b* equal to that of activity *S*.

There can be two kinds of sequences as denoted in Fig. 12: One is the sequence with the same arrival rate as that of the split activity S (Fig. 12(a)). The other is the sequence with arrival rate $p_i\lambda$ compared to the arrival rate λ of the split activity S, where p_i is the selection probability of outgoing branch i (Fig. 12(b)). Let μ' be the service rate of activity b. Thus,

$$\mu' = \frac{1}{\sum_{i=1}^{n} \frac{1}{\mu_i - \lambda}} + \lambda \quad \text{in case of the former}$$

and

$$\mu' = \frac{1}{\sum_{i=1}^{n} \frac{1}{\mu_i - p_j\lambda}} + \lambda \quad \text{in case of the latter.}$$

Here, μ_i is the service rate of activity i within the longest execution path and p_j is the selection probability of the path.

4. An overall example

In this section, we show the overall behavior of our WCP method as in Fig. 13. Let the arrival rate to the workflow schema be 10 and the branch probabilities of OR-split and LOOP control constructs, i.e.,

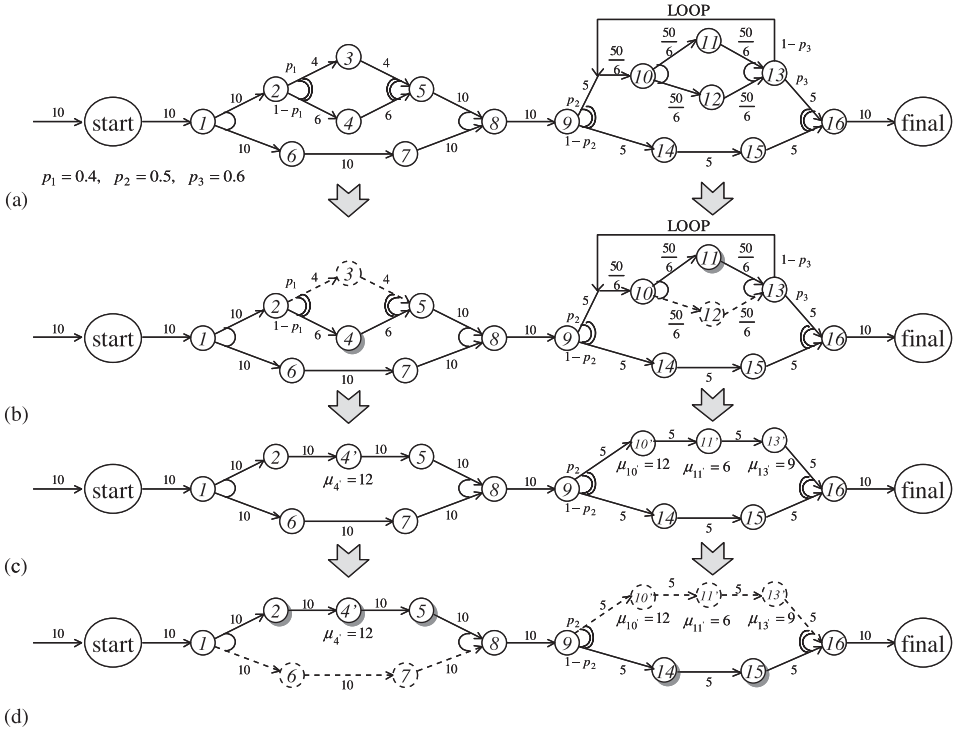


Fig. 13. An overall example of the WCP method.

Table 1
Activities' service rates

Activity	Service rate	Activity	Service rate	Activity	Service rate
1	13	2	15	3	7
4	8	5	20	6	18
7	20	8	14	9	16
10	20	11	10	12	15
13	15	14	6	15	7
16	25				

p_1, p_2, and p_3 be 0.4, 0.5 and 0.6, respectively. Thus, we can specify the arrival and departure rate at each activity as in Fig. 13(a), which is based on Section 2.2. And, the service rate of each activity is given in Table 1.

There are two top-level control blocks in Fig. 13(a), and we should find out the critical sub-path in each control block in order to determine the critical path. As we mentioned in the WCP algorithm, when deciding the critical sub-path in a top-level control block, we select the longest execution path from the innermost to the outermost control block of the top-level control block. Because a (OR-split, OR-join) control block and a (AND-split, AND-join) control block are innermost control blocks in Fig. 13(a), we select the longest execution paths in the control blocks, which are activity 4 and 12, respectively, as denoted in Fig. 13(b). By applying Sections 3.2 and 3.7 to the Fig. 13(b), newly transformed activities $4'$, $10'$, $11'$, and $13'$ can be obtained in Fig. 13-(c). If we determine the longest execution paths from the two atomic control blocks in Fig. 13(c), the critical sub-paths of the two top-level control blocks are selected as in Fig. 13(d). Finally, we can determine the critical path, {1, 2, 4, 5, 8, 9, 14, 15, 16}, in the workflow schema by combining these critical sub-paths.

Next, let us consider the loan processing in a bank that is depicted in Fig. 14. When a loan applicant enters a loan request, the bank first evaluates the risk incurred by the loan and his credit worthiness at the same time. If the evaluation record is acceptable, the bank decides to accept the loan request after investigating the documents submitted by the applicant. If the loan request is finally accepted, the bank notifies the result to the applicant, transfers the loaned money, and logs the processing result. But, if the loan request is rejected, the bank notifies the result to the applicant and logs the processing result. The loan processing of Fig. 14 is composed of 21 activities, each service time of which is specified in Table 2. Our WCP method determines the critical path of the loan processing as a sequence of grayed activities in Figure 14, i.e., {1, 2, 4, 8, 9, 12, 13, 14, 15, 16, 17, 18}.

From the concept of the critical path, the workflow instances passing along the critical path may have high probabilities of missing the workflow deadline. If a workflow instance misses the workflow deadline, special actions, referred to as escalations, may be triggered. Because escalations always result in increased operational costs for workflow processing, it is desirable to reduce the number of workflow instances that result in being escalated as well as the operational costs associated with escalations. The workflow instances passing along the critical path may be highly escalated. And, we can expect that the critical path have the higher escalation rate than other execution paths, which will be shown in the experiment performed by using DEVSim++. DEVSim++ is a C++-based discrete event modeling framework, which has been used in many areas of systems' design such as communication network design and parallel computer architecture design. The loan processing depicted in Fig. 14 has 24 different execution paths.

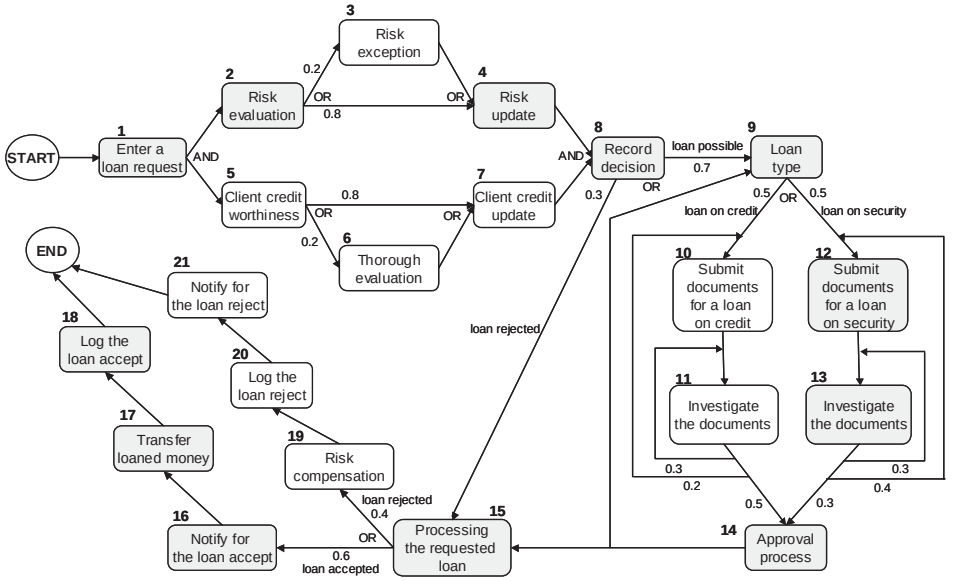


Fig. 14. Loan processing.

Table 2
Loan activities' service times

Activity	Service time (s)	Activity	Service time (s)	Activity	Service time (s)
1	0.1	2	0.3	3	0.2
4	0.1	5	0.1	6	0.2
7	0.2	8	0.2	9	0
10	0.2	11	0.15	12	0.1
13	0.2	14	0.4	15	0
16	0.15	17	0.1	18	0.2
19	0.25	20	0.3	21	0.2

Workflow deadline: 8 s.

Let the sequence of activity {1, 2, 3, 4, 8, 9, 10, 11, 14, 15, 16, 17, 18} be *Path 1*, the sequence of activity {1, 2, 3, 4, 8, 9, 10, 11, 14, 15, 19, 20, 21} be *Path 2*, $\cdots$, and the sequence of activity {1, 5, 6, 7, 8, 15, 19, 20, 21} be *Path 24*. And, let the workflow deadline be 8 s. Table 3 shows the escalation rate of workflow instances for each execution path as a result of the experiment in which we generate 10,000 workflow instances. We can notice that the escalation rate of *Path 9*, which is identified as the critical path by our WCP method, is higher than that of other path. This indicates that the method proposed in this paper is valid.

5. Discussion

Besides workflow control constructs specified in this paper, two synchronization edges (simply sync edges) mentioned in [19] can be considered to support synchronizations of activities from different paths

Table 3
The escalation rates of execution paths

Execution path	# of passed instances	# of escalated instances	Escalation rate (%)
Path 1	211	1	0
Path 2	346	5	1.4
Path 3	240	21	8.7
Path 4	344	31	9
Path 5	177	0	0
Path 6	303	0	0
Path 7	720	13	1.8
Path 8	1141	11	0.9
Path 9	**735**	**78**	**10.6**
Path 10	1184	69	5.8
Path 11	609	0	0
Path 12	993	0	0
Path 13	300	6	2
Path 14	449	4	0.9
Path 15	288	13	4.5
Path 16	435	14	3.2
Path 17	305	0	0
Path 18	407	0	0
Path 19	102	0	0
Path 20	180	2	1.1
Path 21	121	7	5.7
Path 22	172	14	8.1
Path 23	89	0	0
Path 24	149	0	0
Total	10,000	289	

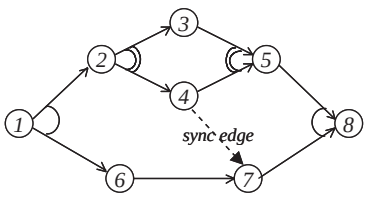


Fig. 15. Synchronization edge.

of a parallel processing as in Fig. 15. Namely, a "soft" sync edge $n_1 \rightarrow n_2$ is used to specify that n_2 may only be executed if n_1 is either completed or if it cannot be triggered anymore. A "strict" sync edge $n_1 \rightarrow n_2$ requires that n_1 must be successfully completed before n_2 is allowed to start.

When there is a sync edge $n_1 \rightarrow n_2$ between activity n_1 and n_2 that belong to different paths of a parallel processing, the average execution time of the path containing activity n_2 (Let us call it $Path(n_2)$) may be affected by that of the path containing activity n_1 (Let us call it $Path(n_1)$). If the average execution time until activity n_2 not including itself at $Path(n_2)$ is less than the average execution time until activity

Table 4
Workflow throughput according to escalation policies

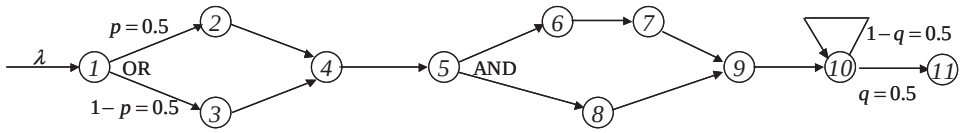
Arrival rate	# of timely completed workflow instances in a workflow with only hard deadlines	# of timely completed workflow instances in a workflow with hard/soft deadlines
2	8005	8322
4	7706	8109
6	7614	7980
8	7517	7928
10	4059	6644

n_1 including itself at $Path(n_1)$, the execution of activity n_2 is delayed until activity n_1 is completely executed. If not, the sync edge has no effect. In consequence, our workflow model can easily support the synchronization edge without no great changes.

In general, an activity is said to be critical only if it belongs to the critical path [4,21]. We can moderate the concept of a critical activity according to the applied applications as follows. An alternative control flow may have more than one OR branches that can belong to the critical path or a path whose total execution time is within distance d from the critical path. If an activity belongs to either the critical path or OR branches within distance d from the critical path, we call it a critical activity; otherwise, a non-critical activity. The value of distance d may be decided by a workflow analyzer. We can expect that a workflow instance missing the deadline of any critical activity has very high possibility not to meet the workflow deadline finally. Hence, if a workflow instance does not meet the deadline of a critical activity, it may be immediately escalated, which indicates that the deadline is hard. On the other hand, the deadline of a non-critical activity is soft in the sense that we make a workflow instance continue to perform its execution if it does not exceed the activity deadline by α percent, where α is the threshold controlled by a workflow analyzer.

Table 4 shows the experimental result of the scheme that distinguishes activity deadlines into two types (i.e., hard and soft) and applies different escalation policies to each type. The experiment is based on the workflow schema of Fig. 16. And, we assume that d is 0 and α is 20%. We can notice in the experimental result that the number of timely completed workflow instances, i.e., workflow throughput is affected by the applied escalation policy. One escalation policy is that workflow instances not to meet any activity deadline are immediately escalated because all activities have only hard deadlines. The other is that activity deadlines can be hard or soft according to the concept of critical and non-critical activities. The escalation policy distinguishing between hard and soft activity deadlines can somewhat reduce the possibility that workflow instances being completed timely in the long run may be escalated due to missing the soft deadlines in any intermediate activities.

Table 4 shows the experimental result of the scheme that distinguishes activity deadlines into two types (i.e., hard and soft) and applies different escalation policies to each type. The experiment is based on the workflow schema of Fig. 16. And, we assume that d is 0 and α is 20%. We can notice in the experimental result that the number of timely completed workflow instances, i.e., workflow throughput is affected by the applied escalation policy. One escalation policy is that workflow instances not to meet any activity deadline are immediately escalated because all activities have only hard deadlines. The other is that activity deadlines can be hard or soft according to the concept of critical and non-critical activities. The escalation policy distinguishing between hard and soft activity deadlines can somewhat reduce the



Activity	Service Time (s)	Deadline (s)	Activity	Service Time s	Deadline s
1	0.05	0.1	7	0.1	2.123
2	0.2	0.561	8	0.1	2.123
3	0.1	0.561	9	0.2	3.115
4	0.2	1.553	10	0.1	4.007
5	0.05	1.653	11	0.2	5
6	0.1	1.888			

Workflow deadline: 5 s

Fig. 16. Activity hard/soft deadlines.

possibility that workflow instances being completed timely in the long run may be escalated due to missing the soft deadlines in any intermediate activities.

6. Conclusion

In this paper, we have first specified the extended workflow control constructs and have defined a well-formed workflow schema. And then, we have proposed a method to determine the critical path in a well-formed workflow schema. Finally, some examples and experiments have been provided to show the overall behavior and the validness of the proposed method. Several workflow structural terms defined in this paper can help design a workflow schema free from some structural and semantic errors. Especially, a nested and a top-level control block with systematic analyses from the innermost to the outermost control block make it easy to apply our critical path algorithm.

We expect that the concept of the critical path can be effectively utilized in many workflow issues, especially workflow resource and time management. In workflow time management, if a workflow instance does not meet the workflow deadline, exception handling called escalation occurs. Because the escalation generally gives high overhead to workflow systems, it is important to minimize the number of escalated workflow instances. The information on the critical path can be utilized for the assignment of workflow and activity deadlines because the real execution times of activities in the critical path directly affects the total workflow completion time.

For further research, we are investigating the way that can verify the structural and semantic correctness of a workflow schema before workflow execution. In particular, it is essential for a mission-critical business process such as E-commerce and financial trading.

References

[1] W.M.P. Aalst, A.P. Barros, A.H.M. Hofstede, B. Kiepuszewski, Advanced workflow patterns, in: The Seventh International Conference on Cooperative Information Systems (CoopIS), September 2000, pp. 18–29.

[2] E.R. Clayton, L.J. Moore, PERT vs. GERT, J. Systems Management (1972) 11–19.

[3] R.L. Disney, Queueing networks, American Mathematical Society Proceedings of Symposium in Applied Mathematics, 1981.

[4] J. Eder, E. Panagos, M. Rabinovich, Time constraints in workflow systems, in: BIS'99 Third International Conference on Business Information Systems,Springer, London, 1999, pp. 265–280.

[5] C. Hagen, G. Alonso, Flexible exception handling in the Opera process support system, in: Proceedings of the 18th IEEE International Conference on Distributed Computing Systems,1998.

[6] P. Heinl, Exceptions during workflow execution, in: Proceedings of the Sixth International Conference on Extending Database Technology,1998.

[7] E. Kafeza, K. Karlapalem, Gaining control over time in workflow management applications, in: Proceedings of the 12th International Conference and Workshop on Database and Expert Systems Applications,2000, pp. 232–242.

[8] C.Y. Kao, An automated scheduling system for project management, in: Proceedings of the 1985 ACM Computer Science Conference, March 1985, pp. 259–270.

[9] B. Kao, H. Garcia-Molina, Deadline assignment in a distributed soft real-time system, in: Proceedings of the 13th International Conference on Distributed Computing Systems,1993.

[10] L. Kleinrock, Queueing Systems: Computer Applications, Wiley, New York, 1974.

[11] P. Lawrence, Workflow Handbook 1997, Wiley, New York, 1997.

[12] F. Leymann, D. Roller, Production Workflow: Concepts and Techniques, Prentice-Hall, Englewood Cliffs, NJ, 1999.

[13] J.J. Moder, R.P. Cecil, W.D. Edward, Project management with CPM, PERT and Precedence Diagramming, third ed., Van Nostrand Reinhold, New York, 1983.

[14] C. Mohan, Recent trends in workflow management products, standards, and research, in: Proceedings of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability, 1997.

[15] S.K. Oh, J.H. Son, Y.J. Lee, M.H. Kim, An efficient method for allocating workflow tasks to improve the performance of distributed workflows, in: International Conference on Computer Science and Informatics, 2000.

[16] E. Panagos, M. Rabinovich, Reducing escalation-related costs in WFMSs, in: Proceedings of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability,1997.

[17] E. Panagos, M. Rabinovich, Predictive workflow management, The Third International Workshop on NGITS, 1997.

[18] H. Pozewaunig, J. Eder, W. Liebhart, ePERT: extending PERT for workflow management systems, The First European Symposium in ADBIS, 1997.

[19] M. Reichert, P. Dadam, $ADEPT_{flex}$-supporting dynamic changes of workflows without loosing control, J. Intell. Inform. Systems 10 (2) (1998) 93–129.

[20] E. Sauls, The use of GERT, J. Systems Management (1972) 18–21.

[21] J.H. Son, M.H. Kim, Improving the performance of time-constrained workflow processing, J. Systems Software 58 (3) (2001) 211–219.

[22] J.H. Son, J.H. Kim, M.H. Kim, Hard/soft deadline assignment for high workflow throughput, in: Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments,1999.

[23] H.A. Taha, Operations Research, Macmillan Publishing Company, New York, 1992.

[24] J.D. Wiest, F.K. Levy, A Management Guide to PERT/CPM: with GERT/PDM/DCPM and Other Networks, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[25] R.W. Wolff, Stochastic Modeling and the Theory of Queues, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[26] Z. Zhu, R.B. Heady, A simplified method of evaluating PERT/CPM network parameters, IEEE Trans. Eng. Management 41 (4) (1994) 426–430.